

引用格式:李炳辰,黄鲁一种移动卷积神经网络的 FPGA 实现[J]. 微电子学与计算机,2019,36(9):7-11. [LI B C, HUANG L. Hardware implementation of a convolutional neural network for mobile terminal based on FPGA[J]. Microelectronics & Computer, 2019, 36(9): 7-11.]

一种移动卷积神经网络的 FPGA 实现

李炳辰,黄 鲁

(中国科学技术大学 微电子学院,安徽 合肥 230026)

摘 要:卷积神经网络是深度学习的一种重要模型,广泛应用于图像处理等领域.常用的神经网络模型因结构复杂,参数众多,不适于放在移动端运行.本文基于模块化和硬件复用的思想,给出了一种基于 FPGA 的手写数字字符识别网络的硬件实现,基于 MobileNet 的原理改进结构,在实现了算法硬件加速的同时,有效地降低了网络的参数数量和整体运算量.基于 MNIST 数据集的实验结果表明,对比传统结构的神经网络,改进结构的参数量减少了 23.26%,计算量减少了 31.32%,在保持速度不变的前提下,用更少的资源和更低的功耗实现了整个网络.

关键词:FPGA;卷积神经网络;硬件加速;MobileNet;移动端

中图分类号: TP183

文献标识码: A

文章编号: 1000—7180(2019)09—0007—05

Hardware implementation of a convolutional neural network for mobile terminal based on FPGA

LI Bing-chen, HUANG Lu

(School of Microelectronics, University of Science and Technology of China, Hefei 230026, China)

Abstract: Convolutional neural networks are an important model of deep learning and are widely used in image processing and other fields. The commonly used neural network model is complex and has many parameters, which is not suitable for running on the mobile end. Based on the idea of modularization and hardware reuse, this paper presents a hardware implementation of handwritten digital character recognition network based on FPGA. Based on the principle of MobileNet, the structure is improved, and the algorithm hardware acceleration is realized, and the number of parameters of the network and the overall calculation amount are effectively reduced. The experimental results based on the MNIST dataset show that compared with the traditional neural network, the parameter size of the improved structure is reduced by 23.26%, and the calculation amount is reduced by 31.32%. The entire network is implemented with less resources and lower power consumption while maintaining the same speed.

Key words: FPGA; convolutional neural network; hardware acceleration; MobileNet; mobile terminal

1 引言

卷积神经网络(CNN)是深度学习的一种重要模型,因其高度的适应性和出色的识别能力,广泛应用于分类识别、目标检测、目标跟踪等领域^[1].目前最常用的 CNN 结构有 VGG^[2]、GoogleNet^[3]、Res-

Net^[4]等.
CNN 的成功崛起得益于自身结构的优势,以及处理器性能的提升. CNN 采用权值共享的结构,降低了模型的复杂度,减少了权值的数量,提高了网络的整体性能^[5];处理器性能的提升,让 CNN 向更深的层次、更大的结构迈进,从而使其能够胜任更多更

复杂的场景.

虽然基于 FPGA 实现 CNN 的相关研究进展很多^[6],但目前 CNN 依然主要部署在 CPU 或 GPU 集群上^[7].传统结构的 CNN 在计算量、存储空间以及能耗方面的巨大需求^[8],令移动和车载终端难以承受,如表 1 所示.

表 1 主流 CNN 结构的计算量和参数量

模型	GoogleNet	VGG16	ResNet50
计算量	1.58G	15.5G	3.86G
参数量	6.99M	138M	25.5M

基于以上原因,一些团队和个人着手进行网络结构小型化的研究工作.其中谷歌提出的 MobileNet 结构^[9],在几乎不影响准确度的情况下,理论上能实现卷积参数量近 90% 的减少,为 CNN 在终端的普及找出了一种实用的技术方案.

2 MobileNet 结构思想原理

MobileNet 的核心思想是使用深度可分离卷积(depth-wise separable convolution)结构来代替传统的 3D 卷积,该结构进一步减少了参数的冗余度,网络的计算量和参数量明显下降,使得卷积网络可以更广泛的应用在移动端平台上.

深度可分离卷积结构的思想如下:

(1) 如图 1(a) 所示,传统卷积结构,假设 M 为输入的通道数, D_F 为输出的宽和高, D_K 为卷积核的宽和高,那么 N 个卷积核处理完全部输入后总的计算量为

$$D_K \times D_K \times M \times N \times D_F \times D_F \quad (1)$$

(2) 如图 1(b) 所示,深度可分离卷积结构,先用一组二维的卷积核进行卷积计算,每个卷积核处理一个输入通道的数据,此步骤的计算量为

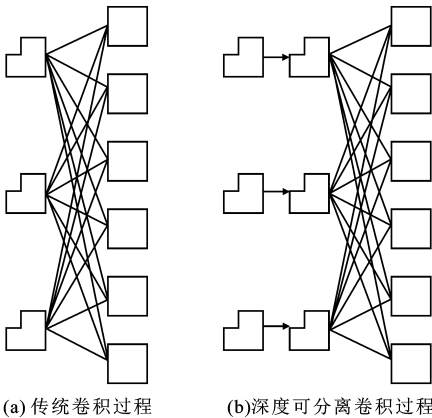


图 1 传统卷积过程和深度可分离卷积过程

$$D_K \times D_K \times M \times 1 \times D_F \times D_F \quad (2)$$

第一步处理完之后,再用一组三维的 1×1 卷积核来处理前一步的结果,最终实现和传统卷积相同的输出,则此步骤的计算量为

$$1 \times 1 \times M \times N \times D_F \times D_F \quad (3)$$

所以深度可分离卷积结构总的计算量为

$$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F \quad (4)$$

(3) 综上可得,深度可分离卷积结构的计算量是传统卷积结构的

$$\frac{D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F}{D_K \times D_K \times M \times N \times D_F \times D_F} = \frac{1}{N} + \frac{1}{D_K^2} \quad (5)$$

根据上述结论可知,卷积网络的通道数越多、卷积核越大,计算量减少的越明显.

3 加速器设计方案

3.1 CNN 结构框架

本文采用的 CNN 结构如下:

Input layer: 28×28 ;

Conv1: 3 kernels, size 5×5 , padding = 0, stride = 1;

Maxpooling1: size 2×2 , stride = 2;

Conv2: 6 kernels, size 5×5 , padding = 0, stride = 1;

Maxpooling2: size 2×2 , stride = 2;

Fc: 10 parameters;

Softmax: classification result;

具体模型如图 2 所示.

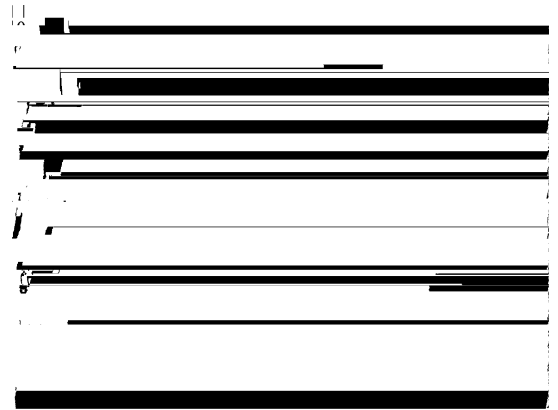


图 2 常规网络结构

图 3 是改进后的结构.因为第一个卷积层的输入通道数是 1,不需要进行改进,所以只把第二个卷积层改成了深度可分离卷积结构.

经过改进,进行单次分类识别时,整个卷积网络的参数量从 1 522 个减少为 1 168 个,减少了

23.26%;计算量从72 960次减少为50 112次,减少了31.32%,效果十分显著.

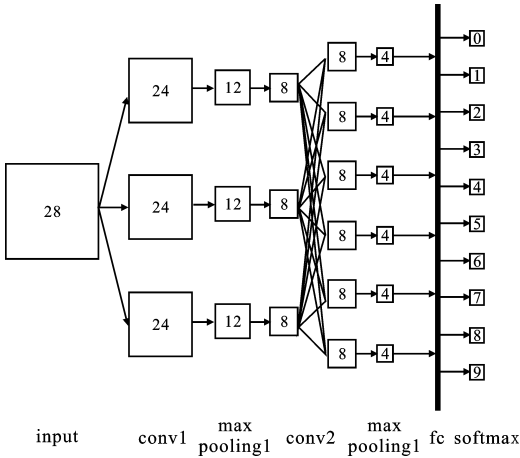


图 3 改进网络结构

3.2 系统硬件结构

本文采用模块化的方式来实现卷积神经网络的总体硬件架构,包括窗口产生模块、卷积计算模块、激活和池化模块、全连接模块以及 softmax 模块. 具体的实现如图 4 所示.

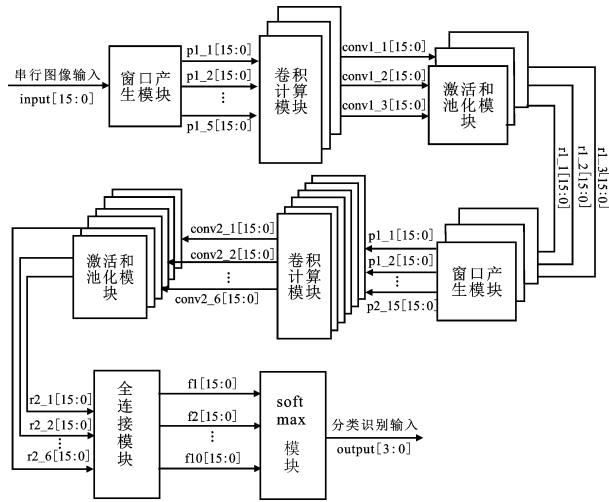


图 4 常规网络硬件框图

图 4 中箭头代表着数据的流向;图中第一行各个模块组成第一层卷积层,第二行各个模块组成第二层卷积层;部分模块堆叠在一起,代表着它们之间是并行处理的关系.

改进结构是在此基础上把第二层的卷积计算模块一分为二,具体变动如图 5 所示.

使用模块化的设计方案,能够根据框架调整,方便快捷的调用相应的模块,而不再需要重新编写整个网络,从而大大减小开发周期. 这将在下面针对各个模块的详细介绍中体现出来.

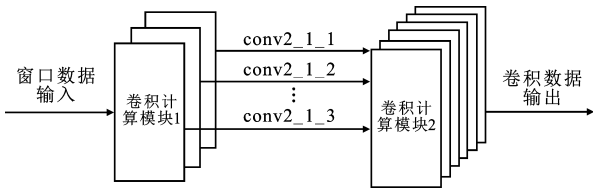


图 5 第二层卷积计算模块一分为二

3.2.1 窗口产生模块

顾名思义,窗口产生模块用来产生卷积滑动窗口. 它主要由 4 位宽为 16 的 FIFO 组成,用以把串行输入的图像数据,变成 5×5 并行窗口数据. 该模块的具体结构如图 6 所示.

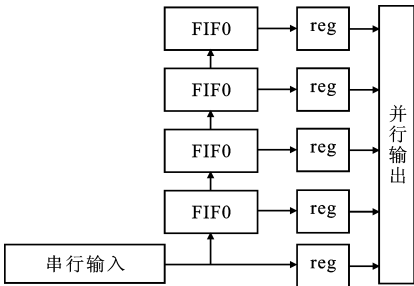


图 6 窗口产生模块

第一层的窗口产生模块中 FIFO 的深度为 28,第二层的窗口产生模块中 FIFO 的深度为 12.

本文把窗口产生模块和卷积计算模块拆开,共有两个好处:一是结构适应性强,改进结构能直接复用模块. 二是改进结构包含大量 1×1 的卷积计算,其计算过程和全连接计算相同,把窗口模块和计算模块分开,就能复用 1×1 卷积计算模块和全连接计算模块,进一步简化了工作.

3.2.2 卷积计算模块

卷积计算模块是整个网络的核心模块. 图 7 是单个卷积核的内部结构. 该结构由 25 个乘法器和 24 个加法器组成,每个时钟周期输入 5×5 都能输出一个卷积计算结果.

该结构的优点很明显,串行处理器实现一个 5×5 的卷积计算至少需要 49 个周期,该结构只需一个时钟周期,利用面积换速度的方法实现了硬件加速.

在 Conv2 层,数据在到来之前经过了一个池化层,所以该模块每两个时钟周期才到来一组窗口数据. 本文将到来的数据平分到这两个周期中计算,硬件采用分时复用的方式,既不会对网络的运行速度造成影响,又节约了一半的乘法器.

经过计算,该方案在常规网络中节约了 225 个乘法器,在改进网络中节约了 46 个乘法器.

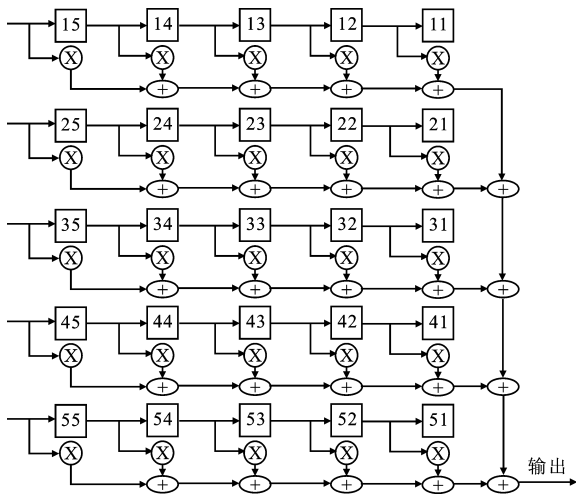


图7 卷积计算模块

3.2.3 激活函数和池化模块

本文采用最大池化,并用 relu 函数作为激活函数,relu 的表达式如下:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (6)$$

从式(6)可以看出,一个比较器就能完成激活运算.因此本文将激活函数和最大池化放到了一个模块里,用3个比较器完成了激活池化运算.

3.2.4 全连接模块

全连接模块也是重点设计的一个模块.全连接层要到来 $4 \times 4 \times 6$ 一共 96 个数据,每个数据需要进行 10 次乘法运算.文献[10]使用了 960 个乘法器来完成这些运算.

但仔细分析数据流就能发现,数据在到达全连接层之前经过了两个池化层,因此每 4 时钟个周期到来一组数据,分 4×4 次到来,即全部数据需要 64 个周期才能全部到来.文献[11]使用的 960 个乘法器在 64 个周期中只计算一次,因此存在很大的优化空间.本文基于数据流的这种规律,设计了图 8 所示的全连接模块,一共使用 15 个乘法器,采用分时复用的方式,在全部的 64 个周期都进行处理批次到来的数据,在不影响整个网络的运行速度的前提下,用 $1/64$ 的乘法器达到了和文献[10]同样的效果,极大地节省了资源.

3.2.5 softmax 模块

几乎所有多分类(识别种类在两个以上)网络的最后都是 softmax 层,softmax 的公式如下

$$P(j) = \frac{e^{k_j}}{\sum_{i=0}^n e^{k_j}} \quad (7)$$

对于 n 分类网络,一个前项传播过程要输出 n

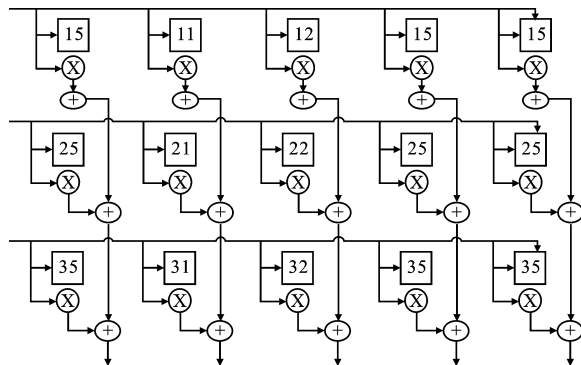


图8 全连接模块

个结果,分别和 n 个种类一一对应.softmax 的作用就是把这些输出归一化为 $[0,1]$ 之间的实数,让它们的和为 1,即概率之和为 1.softmax 层能很好的提高反向传播时的训练效率,但是在前向传播时则没有必要.实质上前向传播分类时,只需找出值最大的那一项就能确定网络最终的输出,因此本文用比较器阵列取代了复杂的指数运算,实现最终的输出.

4 实验结果及分析

4.1 实验环境

本文实验使用 Xilinx 公司 Spartan6 系列 xc6slx45 芯片作为硬件平台,使用 ISE14.7 软件和 Verilog 语言进行开发和测试.

本文实验采用 MNIST 数据集作为手写数字字符识别的输入图像^[11],该数据集包含 60 000 张训练图像和 10 000 张测试图像,每张图像为单通道灰度图像,大小为 28×28 ,每个像素的取值在 $0 \sim 255$ 的整数范围,在实验进行前,对该数据集进行了初始化,使得输入的数值范围变为 $[0,1]$.

在通用 CPU 平台上用软件实现了与 FPGA 完全一致的 CNN 模型,用于对比. CPU 采用 Intel Core i5-7500 四核处理器,主频为 3.4 GHz.软件开发环境是在基于 Python 的 Keras 库,该环境专门针对简易和快速的原型设计进行了优化.

FPGA 上只完成前向传播过程,即只分类,不训练.整个网络是在 CPU 上用 Keras 来训练的,训练完成后,将权重进行 16 位定点化并下载到 FPGA 板上 RAM 中,供 FPGA 使用.

4.2 实验结果

表 2 列出了 FPGA 的资源使用情况.本文通过资源复用,用更少的乘法器(315 对 1 485)和 Register 资源达到了文献[10]同样的效果;由于本文使用的 FPGA 中的 DSP 资源较少,需要用 LUT 综合出

乘法器,所以使用了较多的 LUT 资源.

表 2 FPGA 资源消耗对比

模块	资源使用数		
	文献网络	本文常规网络	本文改进网络
Registers	64 364	4 665	4 775
LUTs	10 078	14 089	10 355
RAM	23	27	27
DSPs	1 485	58	58

表 3 是不同平台不同结构下 CNN 运行速度的对比. 本文常规网络在 CPU 上运行了 61.33 秒,约是文献[10]的两倍,速度差距与 CPU 性能差距相同. 本文 FPGA 的工作频率为 50 MHz,与文献[10]相同,但本文方案在 FPGA 上运行的时间更短. 改进网络在 FPGA 上的运行速度和常规网络一致,与预期相符.

表 3 不同平台不同结构下 CNN 的运行速度对比

平台	图像/迭代	时间/s
参考文献 CPU	60 000	34.02
参考文献 FPGA		4.33
本文 CPU & 常规网络		61.33
本文 CPU & 改进网络		97.52
本文 FPGA & 常规网络		2.37
本文 FPGA & 改进网络		2.53

本文使用 Xilinx 公司提供的 Xilinx Power Estimator(XPE)工具来估测 FPGA 芯片的功耗,具体表现如表 4. 文献[10]的功耗达到了 60 W,本文的两种网络都不到 2 W,这里只对比本文的两种网络. 可以看到,在只改进一层卷积层的情况下,改进网络就有 6.74%的功耗降低,将应用到更大规模的网络上将有更好的表现,这对将神经网络移植到移动设备上有重大意义.

表 4 FPGA 功耗对比

平台	功耗/W
参考文献	60
常规网络	1.78
改进网络	1.66

综上,本文所提出的网络,在资源消耗、运行速度和能耗方面都有着优于文献[10]的表现.

5 结束语

本文提出了一种基于 FPGA 的卷积神经网络硬件实现方案. 实验结果表明,针对输入大小为 28 * 28 的手写数字字符识别,该方案充分发挥 FPGA 硬件平台的并行执行优势,并且兼顾硬件资源的复用. 相比参考文献,该方案充分挖掘 CNN 的结构特点,优化数据流程,复用硬件电路,以更少的资源和更低的功耗实现了更快的分类识别. 该方案采取模块化设计,对 FPGA 实现卷积神经网络得加速具有一定的通用性和重要意义.

参考文献:

[1] YU K, JIA L, CHEN Y, et al. Deep learning: yesterday, today, and tomorrow[J]. Journal of computer Research and Development, 2013, 50(9): 1799-1804.

[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. Computer Science - Computer Vision and Pattern Recognition, arXiv preprint arXiv:1409.1556, 2014.

[3] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]// Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, IEEE, 2015: 1-9.

[4] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, IEEE, 2016: 770-778.

[5] 陈雯柏. 人工神经网络原理与实践[M]. 西安:西安电子科技大学出版社, 2016.

[6] LACEY G, TAYLOR G W, AREIBI S. Deep learning on fpgas: past, present, and future[J]. arXiv preprint arXiv:1602.04283, 2016.

[7] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: convolutional architecture for fast feature embedding[C]// Proceedings of the 22nd ACM international conference on Multimedia, Orlando, Florida, ACM, 2014: 675-678.

[8] QIU J, WANG J, YAO S, et al. Going deeper with embedded fpga platform for convolutional neural network[C]// Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, ACM, 2016: 26-35.

[9] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[10] 方睿, 刘加贺, 薛志辉, 等. 卷积神经网络的 FPGA 并行加速方案设计[J]. 计算机工程与应用, 2015, 51(8): 32-36.

[11] LECUN Y L, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

作者简介:

李炳辰 男, (1995-), 硕士. 研究方向为 FPGA 算法加速.
E-mail: hnlibc@mail.ustc.edu.cn.

黄鲁 男, (1961-), 硕士, 副教授. 研究方向为数模混合高速接口集成电路设计.